International Journal of Advanced Trends in Engineering and Technology (IJATET)
Impact Factor: 5.665, ISSN (Online): 2456 - 4664
(www.dvpublication.com) Volume 2, Issue 1, 2017

SURVEY OF REDUNDANT BASED FAULT TOLERANT TECHNIQUES FOR EMBEDDED SYSTEM

P. Loganath
Assistant Professor, Electronics and Communication Engineering, SSM College of Engineering, Komarapalayam, Tamilnadu

Cite This Article: P. Loganath, "Survey of Redundant Based Fault Tolerant Techniques for Embedded System", International Journal of Advanced Trends in Engineering and Technology, Page Number 66-71, Volume 2, Issue 1, 2017.

Abstract:
With continued scaling of silicon process technology, producing reliable electronic components in extremely denser technologies pose a challenge. Further, the systems fabricated in deep sub-micron technology are prone to intermittent or transient faults, causing unidirectional errors, upon exposure to ionizing radiations during system operation. The ability to operate in the intended manner even in the presence of faults is an important objective of all electronic systems. In order to achieve fault-tolerance, each module of the system must be fault-tolerant by possessing run-time (or online) fault detection capabilities. Totally self-checking (TSC) circuits permit online detection of hardware faults. The objective of this project is to analyze the area (resource utilization), speed and power consumption for Look-Up Table (LUT) implementation with and without fault detection capability using DMR, TMR on Xilinx FPGA. Triple modular redundancy (TMR) is a technique commonly used to provide design hardening. We apply DMR and TMR technique on n-bit input LUT.

Key Words: Fault Tolerance, FPGA, LUT, DMR & TMR

Introduction:

In recent times due to technological advancements, the performance of integrated circuits has increased a lot also lower device sizes, low power consumption have also helped to improve the performance of the devices. But on the other hand, modern devices have become more susceptible to transient faults and when these faults are executed, it creates soft errors. So soft errors are errors which are not consistent rather they are random. Although Soft errors cannot damage the physical hardware of the chip however they can corrupt the value stored in the chip. Hard errors are related to the system hardware. So the difference between soft errors and hard errors is that, soft errors can be corrected by applying different techniques where as to rectify hard errors physical changes has to be done on hardware. Soft errors can occur due to environmental conditions such as radiation flux, alpha particles, cosmic rays, power supply fluctuations, temperature, pressure, humidity and electromagnetic interference. The errors occurring in an electronic circuit can be broadly classified as symmetric, asymmetric, and unidirectional errors. The error is symmetric if both 0 to 1 and 1 to 0 transitions occur simultaneously in a data word. If only 0 to 1 or 1 to 0 transitions are likely, and the error type is known a priori, then the errors are asymmetric. If both 0 to 1 and 1 to 0 transitions can occur in data words, but in any particular word all errors are of one type, then the errors are called unidirectional errors. The ability to operate in the intended manner even in the presence of faults is an important objective of all electronic systems. In order to achieve fault-tolerance, each module of the system must be fault-tolerant by possessing concurrent fault detection capabilities. Fault detection methods can be broadly classified as Built-In Self-test, roving technique, redundancy technique, logic implications technique and error coding technique. Redundancy is based on either modular redundancy or time redundancy. In modular redundancy, the functional module is replicated two or three times. In time redundancy, the same function is performed by the same functional module more than once. Any difference in these outputs indicates a fault. It is obvious that there is an area overhead or latency overhead by two or three times when using redundancy techniques. Faults can be detected by verifying the code with binary data. Many unidirectional error detecting codes like Parity code, Hamming code, Reed Solomon code, Berger code and Bose code. Berger code can detect all multiple unidirectional errors. Self-checking circuit using Berger code can have Berger encoder implemented as a sequential circuit or as a combinational circuit. The sequential circuit implementation requires more resource overhead to implement counter circuits and takes multiple clock cycles to detect the error. The combinational circuit implementation takes more hardware latency.

System Design:

The proposed system is an analysis of fault tolerant techniques for embedded architectures with different n bit LUT. In order to achieve fault-tolerance, each module of the system must be fault-tolerant by possessing run-time (or online) fault detection capabilities. Totally Self-checking (TSC) circuits permit online detection of hardware faults. The objective of this project is to analyze the area (resource utilization), speed and power consumption for Look-Up Table (LUT) implementation with and without fault detection capability using DMR and TMR on Xilinx FPGA. A LUT (Lookup table) is a one bit wide memory array. LUT's can be programmed and reprogrammed to change the logical function implemented. A Double modular redundancy (DMR) is a technique commonly used self check point. DMR also has the limitation that it can only detect errors, and requires other circuitry to handle recovery after an error is detected. Double modular redundancy (DMR) structure uses

asynchronous C element to output and keep the correct value of two redundant storage cells. A Triple modular redundancy (TMR) is a technique commonly used to provide design hardening. The TMR combined with Single Event Upset (SEU) correction through partial reconfiguration is a powerful and effective SEU mitigation strategy. Each TMR approach will be evaluated in terms of the number of sensitive configuration bits, the number of sensitive configuration bits relative to the amount of logic used, the overhead required, and the speed at which the design can run. This method is only supported for the Virtex TM series of Xilinx FPGAs. Xilinx Application Note, XAPP216, describes the use of Read back and Partial Configuration for SEU detection and correction. The proposed system consists of three modules.

- ✓ Look Up Table
- ✓ Look Up Table with DMR
- ✓ Look Up Table with TMR.

Look Up Table:

The block diagram of LUT is shows in figure 1. It consists of 4 D latch and one 4:1 multiplexer. The LUT can be implemented with the cascade connection of D latch and one 4:1 multiplexer. The LUT output F depends upon the selection lines S_1 S_0. A 4:1 multiplexer receives information from the D latch. During programming, the bit stream is configured in D latch. When PROG signal is low, the content of D latch is configured to the output of LUT.

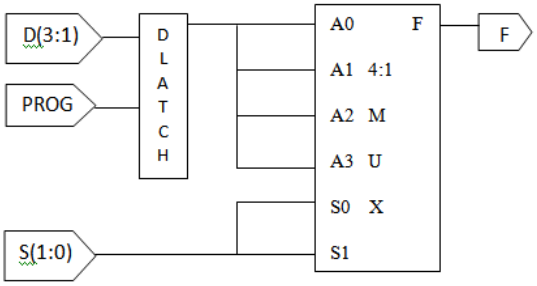

Figure 1: Block Diagram of LUT

The D latch receives that designation from its ability to hold data in its internal storage. A 4:1 MUX receives information from the D latch. Based on the selection lines of MUX (i.e.,) When S_1 S_0 = 00, the first D latch A0 is selected and routed to the MUX output (F). Similarly, When S_1 S_0 = 01, the second D latch A1 is selected and routed to the MUX output (F). When S_1 S_0 = 10, the third D latch A2 is selected and routed to the MUX output (F). When S_1 S_0 = 11, the fourth D latch A3 is selected and routed to the MUX output (F). When PROG signal is high, the content of D latch is configured to the high impedance output F. The following function table 1 explains the operation of LUT.

Table 1: Function table of LUT

Control Signal	**Selection Lines**	**Output**
PROG	**S_1S_0**	**F**
0	xx	Z
1	00	D_0
1	01	D_1
1	10	D_2
1	11	D_3

Double Modular Redundancy:

A machine which is Double Modular Redundancy (DMR) has duplicated elements which work in parallel to provide one form of redundancy. A typical example is a complex computer system which has duplicated notes, so that if one node fails, another is ready to carry on its work. A LUT fault-tolerant machine uses replicated elements operating in parallel. At any time, all the replications of each element should be in the same state. The same inputs are provided to each replication, and the same outputs are expected. The outputs of the replications are compared using the comparator circuit. A machine with two replications of each element termed as Double Modular Redundancy. The comparator circuit can then only detect mismatch and recovery relies on other methods. The comparator circuit can determine which replications is in error and is observed. If the outputs are equal, there is no error. Hence the content of D latch is transferred to the output. The comparator produces an error; the tri-state buffer goes to high impedance. The block diagram of LUT with DMR is shown in Figure 2. It consists of two LUT, one comparator and one tri state. During the programming, the bit stream is configured in D latch. The comparator is a combinational circuit which is used to compare the output of two LUT. If the outputs are equal, there is no error. The comparator produces an error, the tri-state buffer goes to high impedance.

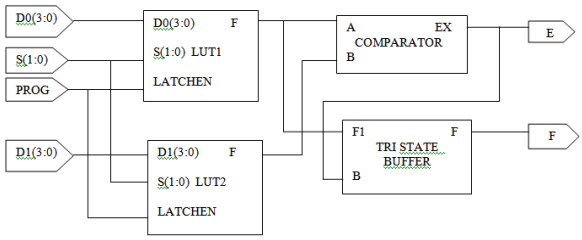

Figure 2: Block Diagram of LUT with DMR

The following function table 2 explains the operation of LUT with DMR. For Example: If both LUTs input is (D0 = 1000 & D1 = 1000) equal, the comparator produces output as 0, so there is no error occurred in the output of the circuit. If both LUTs input is (D0 = 1000 & D1 = 1100) equal, the comparator produces output as 1, so error is occurred in the output of the circuit and its goes to high impedance.

Table 2: Function table of LUT with DMR

LUT-1	LUT-2	Comparat or Output	Output
D0 = 1000	D1 = 1000	0	No error
D0 = 1000	D1 = 1100	1	High impedance

Triple Modular Redundancy:

Three identical look up table are used to compute the specified Boolean function. The set of data at the input of the first circuit are identical to the input of the second and third LUT. In computing, triple module redundancy (TMR) is a fault-tolerant form of N-modular redundancy, in which three systems perform a process and that result is processed by a majority voting system to produce a single output. If any one of the three systems fails, the other two systems can correct and mask the fault. The TMR concept can be applied to many forms of redundancy, such as software redundancy in the form of N-version programming. Some ECC memory uses triple module redundancy hardware (rather than the more common Hamming code), because triple module redundancy hardware is faster than Hamming error correction hardware. Space satellite systems often use TMR, although satellite RAM usually uses Hamming error correction. This technique scheme uses three identical look up table performing the same task in parallel with corresponding outputs being compared through a majority voter circuit. The most common example of TMR is a three LUT with d-type latch that has been triplicated and to which a voter has been added on its output. The block diagram of LUT with TMR is shown in figure It consists of three LUT, one voter and one tri state. During the programming, the bit stream is configured in D latch. The voter is used to produce single output of three LUT. If the outputs are equal, there is no error. Otherwise, the tri-state buffer goes to high impedance.

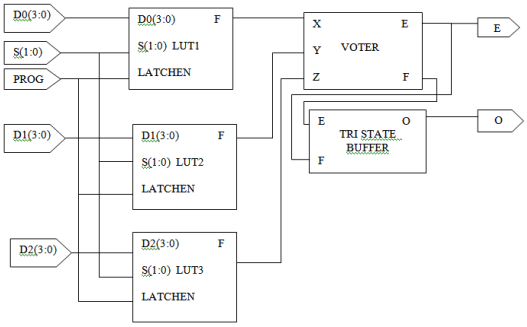

Figure 3: Block Diagram of LUT with TMR

The following function table 3 explains the operation of LUT with TMR. For Example: If all LUTs input is (D0 = 1000 & D1 = 1000 & D2 = 1000) equal, the voter produces output as 0, so there is no error occurred in the output of the circuit. If all LUTs input is (D0 = 1000 & D1 = 1100 & D2 = 1010) equal, the voter produces output as 1, so error is occurred in the output of the circuit and its goes to high impedance.

Table 3: Function table of LUT with DMR

LUT-1	LUT-2	LUT-3	Voter Output	Output	
				E	O
D0 = 1000	D1 = 1000	D2 = 1000	0	0	No Error
D0 = 1000	D1 = 1100	D2 = 1010	1	1	High Impedance

Software Requirement:

Xilinx 9.1i VHDL language

Software Description:

XILINX ISE:

Xilinx Integrated Software Environment is a software tool produced by Xilinx for synthesis and analysis of HDL designs, enabling the developer to synthesize ("compile") their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. Using third-party design-entry software, the designer creates a net list that forms the input to the Xilinx software. Utility software (pin2xnf for Future Net DASH and win2xnf for View logic, for example) translate the net list into Xilinx net list format (XNF) file. In the next step the next step the Xilinx program xnfmap takes the XNF net list and maps the logic into the Xilinx Logic Cell Array (LCA) architecture. The output from the mapping step is a MAP file. The schematic MAP file may then be merged with other MAP files using xnfmerge. This technique is useful to merge different pieces of a design, some created using schematic entry and others created, for example, using logic synthesis. A translator program map21ca translates from the logic gates (NAND gates NOR gates, and so on) to the required CLB configurations and produces an unrouted LCA file. The Xilinx place-and-route software (apr or ppr) takes the unrouted LCA file and performs the allocation of CLBs and completes the routing. The result is a routed LCA file. A control program xmake (that works like the make program in C) can automatically handle the mapping, merging, and place-and route steps. Following the place-and-route step, the logic and wiring delays are known and the post layout net list may be generated. After a post layout simulation the download file or BIT file used to program the FPGA (or a PROM that will load the FPGA) is generated using the Xilinx make bits program. Xilinx also provides a software program (Xilinx design editor, XDE) that permits manual control over the placement and routing of a Xilinx FPGA. The designer views a graphical representation of the FPGA, showing all the CLBs and interconnect, and can make or alter connections by pointing and clicking. This program is useful to check an automatically generated layout, or to explore critical routing paths, or to change and hand tune a critical connection, for example. Xilinx uses a system called X-BLOCK for creating structures such as vectored instances and data paths. This system works with the Xilinx XNF net list format. Other vendors, notably Actel and Altera, use a standard called Relationally Placed Modules (RPM), based on the EDIF standard.

Synthesis and Simulation Design for XILINX:

The Synthesis and Simulation Design Guide provides a general overview of designing Field Programmable Gate Arrays (FPGA) devices with Hardware Description Languages (HDLs). It includes design hints for the novice HDL user, as well as for the experienced user who is designing FPGA devices for the first time. Before using the Synthesis and Simulation Design Guide, you should be familiar with the operations that are common to all Xilinx tools.

VHDL Language:

VHDL is a hardware description language that is used to describe the behavior and structure of digital systems. The acronym VHDL stands for VHSIC Hardware Description Language, and VHSIC in turn stands for Very High Speed Integrated Circuit. However, VHDL is a general purpose hardware description language which can be used to describe and simulate the operation of a wide variety of digital systems, ranging in complexity from a few gates to an interconnection of many complex integrated circuits. VHDL was originally developed to allow a uniform method for specifying digital systems. The VHDL language became an IEEE standard in 1987, and it is widely used in industry. IEEE published a revised VHDL standard in 1993. VHDL can describe a digital system at several different levels-behavioral, data flow and structural. For example, a binary adder could be described at behavioral level inters of its function of adding two binary numbers, without giving any implementation details. The same adder could be described at the data flow level by giving the logic equations for the adder. Finally, the adder could be described at the structural level by specifying the interconnection of the gates which make up the adder. VHDL leads naturally to a top-down design methodology in which the system is first specified at a high level and tested using a simulator. After the system is debugged at this level, the design can gradually be refined, eventually leading to a structural description which is closely related to the actual hardware implementation. VHDL was designed to be technology independent. If a design described in VHDL and implementation in today's technology, the same VHDL description could be used as a starting point in some future technology. The synthesis tool will derive a hardware implementation from the VHDL code.

Hardware Requirements:

- ✓ Spartan-3 FPGA

Hardware Description:

The Spartan-3 families of Field-Programmable Gate Arrays are specifically designed to meet the needs of high volume, cost-sensitive consumer electronic applications. Spartan-3 FPGAs are ideally suited to a wide range of consumer electronics applications, including broadband access, home networking, display/projection and digital television equipment. The Spartan 3 Development Kit provides a platform for engineers designing with the Xilinx Spartan 3 FPGA. The board provides the necessary hardware to not only evaluate the features of the Spartan 3 but also to implement complete user applications. The Spartan-3 Development board was

designed to support the Spartan-3 FPGA in the 676-pin, BGA package (FG676). The FG676 package supports three mid-range densities (1000, 1500, and 2000). The board was designed to support two of the three densities: the 3S1500 and 3S2000. The schematic symbol used for the Spartan-3 device indicates the specific I/O pins available in each density (396 I/Os with 2VP7 and 556 I/Os with the 2VP20/30). The Spartan-3 Development board supports Boundary-scan as well as Master/Slave Serial and Master/Slave Parallel (Select MAP) using the on-board PROMs. The Platform Flash PROM(s) provide easy-to-use non- volatile storage for the configuration file. These devices are in–system programmable via the boundary scan chain and may program the FPGA in Master Serial, Master Select MAP, Slave Serial, or Slave Select MAP modes. The Spartan-3 Dev Board includes two standard 6-pin Mini- Din (PS2) connectors labeled JS1 and JS2. The Spartan-3 Development board is populated with 32MB DDR SDRAM, 16MB Flash, and 2MB SRAM. Additional memory including Flash, SDRAM, and SRAM are available. The Spartan-3 Development Kit includes a 5V AC/DC Adapter that plugs into the board at "J7". The Spartan-3 Development board uses a 5V AC/DC adapter (supplied with the kit) with center positive barrel connector

Implementation Result:

The implementation result is shown in the table 4 and 5.

Table 4: Timing Analysis

BIT LENGTH	LUT (ns)	DMR (ns)	TMR (ns)
4	6.141	7.871	7.936
8	9.383	10.935	10.721
16	9.443	11.422	11.238
32	10.068	11.422	12.359
64	11.307	7.871	7.936

Table 5: Area Analysis

BIT LENGTH	LUT*	DMR	TMR
4	4	9	14
8	5	11	17
16	10	21	32
32	10	21	32
64	21	45	68

Note: * No of 4 input LUT

Device Type: xc3s400-5pq208

Simulation Output:

The simulation output is shown in the figure 4, 5 and 6. The figure 4 shows the simulation output of 4 bit LUT: When latchen=0, output is 0, i.e. f=0 irrespective of any input condition. When latchen=1, the circuit works as follows: If S_1S_0=00, output is 1, i.e. f=1 d3 data is transferred to output. If S_1S_0=01, output is 1, i.e. f=1 d2 data is transferred to output. If S_1S_0=10, output is 1, i.e. f=1 d1 data is transferred to output. If S_1 S_0=11, output is 1, i.e. f=1 d1 data is transferred to output.

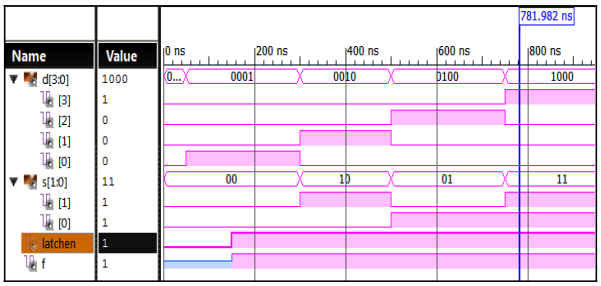

Figure 4: Simulation Output of 4 Bit LUT

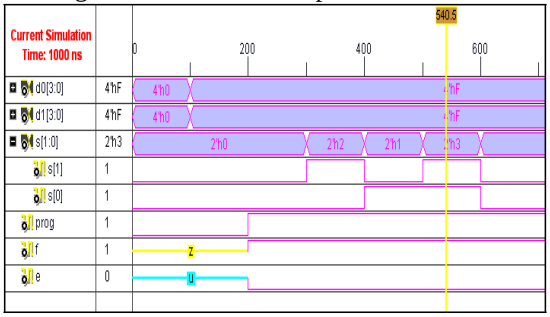

Figure 5: Simulation Output of 4 Bit DMR with LUT

The figure 5 shows the simulation output of 4 bit DMR with LUT: When PROG=0, output is 0, i.e. f=0 irrespective of any input condition. When PROG =1, the circuit works as follows: If both LUTs input is (D0 = 1000 & D1 = 1000) equal, the comparator produces output as 0, so there is no error occurred in the output of the circuit. If both LUTs input is (D0 = 1000 & D1 = 1100) equal, the comparator produces output as 1, so error is occurred in the output of the circuit and its goes to high impedance.

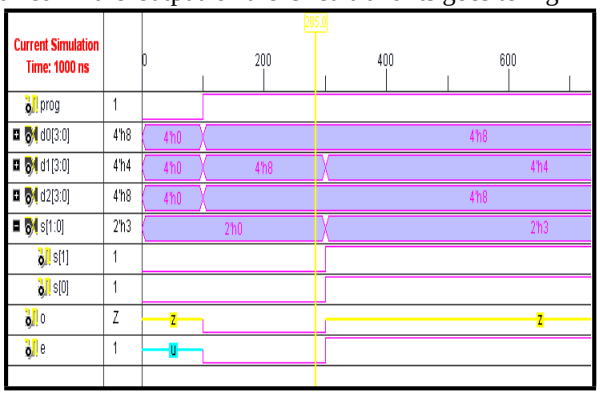

Figure 6: Simulation Output of 4 Bit TMR with LUT

The figure 6 shows the simulation output of 4 bit TMR with LUT: When PROG=0, output is 0, i.e. f=0 irrespective of any input condition. When PROG =1, the circuit works as follows: If any two or three LUTs input is (D0 = 1000 & D1 = 1000 & D2 = 1000) equal, the voter circuit produces output as 0, so there is no error occurred in the output of the circuit. If any two or three LUTs input is (D0 = 1100 & D1 = 1000 & D2 = 1010) different the voter circuit produces output as 1, so error is occurred in the output of the circuit and its goes to high impedance.

Conclusion:

As technology advances, the problem of soft errors is spreading widely. So some techniques are required which can reduce the existing soft errors and increase the performance as well as reliability of a system. Proposed method uses the technique of DMR and TMR on Xilinx FPGA which can detect the soft error very efficiently and it is very less time consuming as well as cost hence the fault- tolerance, of each module of the system must be fault- tolerant by possessing run-time (or online) fault detection capabilities. Totally Self-checking (TSC) circuits permit online detection of hardware faults.

References:

1. Srinivasan,V, J. W. Farquharson, Student Member, IEEE, W. H. Robinson, Member, IEEE, B. L. Bhuva, Senior Member, IEEE,(2006) ,Evaluation of Error Detection Strategies for an FPGA- Based Self-Checking Arithmetic and Logic Unit.
2. Natarajan Somasundaram, Jeong A Lee, Farha Mehdipour, Ramadass Narayanadass, Y V Ramana Rao, (2013),'Scalable Error Detection Coding© Algorithm for Totally Self-Checking (TSC) Circuits SEDC© Algorithm for TSC Circuits" Consumer Electronics Times, Vol. 2 Iss. 3, PP. 116-123.
3. Pradhan. D. K., and J. J. Stiffler, (1980) "Error- Correcting Codes and Self- Checking Circuits", in Computer, vol. 13, no. 3, pp. 27-37.
4. Carl Carmichael, (2001) "Triple Module Redundancy Design Techniques For Virtex FPGAs," Technical report, Xilinx Corporation, XAPP197 (v1.0)